

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/18/2022 | Thomas Bartlett | Initial version of document. |
| 1.1 | 10/01/2022 | Thomas Bartlett | Updated Evaluations Section |
| 1.2 | 10/16/2022 | Thomas Bartlett | Updated Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is wanting to take their app Draw It or Lose It, which is only available as an Android app currently, and make it available as a web-based game. The game will be able to be played across multiple platforms, as users will not all have the same platform they are accessing the game from. The game will support one or more teams playing, each team can have multiple players linked to it. Each game and team will have a unique ID or name, and the game will have only one instance existing at any time.

## [Design Constraints](#_2et92p0)

* Provide a product that is cross-platform and accessible from multiple operating systems.
* The game is available currently as an app on the Android OS. Keeping the user interface identical, or very close to it, will be a benefit to the end users.
* Provide a product that has the ability to support more than one team with each team having multiple players. This will require making sure the server can handle this.
* Provide a product that has only one game in existence in memory at any time. This means having unique identifiers for each game, team, and player.
* Make sure the images currently being used from The Gaming Room’s large library of stock drawings for the Android app are compatible with all of the other platforms.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram is a visual representation of how all the components of the program are connected. In the diagram you can see that the Entity class has a parent/child relationship with the Game, Team and Player classes. This allows the classes to inherit from the Entity class. There are also associations that can be seen. The first is between the GameService class and the Game class. There is a 0 to many association which means that there can be 0 to many games. There is also a 0 to many association between the Game class and the Team class. This shows that for every Game, there can be 0 to many teams. In the same fashion, there is a 0 to many association between the Team and Player classes, showing that for each team there can be 0 to many players. The way that these classes are all related to each other, with inheritance and as an association, is set up to take advantage of the object oriented programming benefits of inheritance and polymorphism. You can also see that the classes also have both public and private attributes and methods, which takes advantage of encapsulation.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The cost to purchase a server for IOS is relatively cheap at only $20. With this it includes macOS Server resources to help with getting the server up and running for development. A down side is that there is not a large population that uses Apple as their main OS outside of the mobile services. | The Linux license beats even the macOS in price as it is free. However, to support development additional resources will be needed, which can increase the cost significantly, and would require research ahead of time as to what resources would be needed to come up with the cost and not waste money on unneeded products. Another downside is the fact that there are not too many people who are familiar with how to navigate and utilize the Linux interface. | Looking at the Windows server, it’s cost is either $20/month or you can out right own it for $972 for their standard edition. The other tier that they offer is their datacenter edition which is $125/month or $6155 to purchase outright. With this being the most expensive of the three operating systems upfront, it is also the most commonly used OS of the three. | An app hosting server costs around $70 - $320 per month. The cost depends on the content, active users, and growth. If you can generate a lot of active users for the app, it can easily cover the cost of the server. |
| **Client Side** | The cost for the server is rather inexpensive, and with the resources provided would not take much time to get up and running. Mac also is not a system that would take much expertise to learn, but again, not too many people use the macOS as their primary system. | With Linux being an open source, costs are significantly lower than that of Apple or Windows. This, however, leads to requiring the client to have prior experience with or learn the OS, which could take time and delay projects, as very few people actually use this OS as a primary system. | Initial costs will be higher for a windows server. However, a large majority of people are very familiar with using Windows that it wouldn’t take long from a learning standpoint, or ease of use. | With mobile devices, the cost can be relatively low if you have a lot of traffic coming to your app. With mobile development, some expertise would be needed, so you would need to take that into consideration. |
| **Development Tools** | For this, a Mac Book can be used with iCode. Coding would then be done using swift. | With most Linux products, Python is pre-installed to be able to use. You would then be able to use IntelliJ Ultimate with Linux as well for coding. | One of the more comprehensive options for a Windows application would be Visual Studio. This allows you to pick from a large variety of languages to code in, including C++ which is what most Windows programs are written in. | When considering mobile development, as stated before, more expertise is needed. You would need to have someone who can develop in the mobile OS specific areas such as Android Studio, or use swift for iPhone app development. You could also go the route of using Unity which supports cross-platform development. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: MY recommendation would be to utilize the Windows OS. Of the available platforms, Windows is the most widely used. It will also help with handling any user requests, provide resources for troubleshooting, and maintaining applications on the back end.
2. **Operating Systems Architectures**: Windows OS has a very simple graphical user interface (GUI) that allows for easy navigation. Another advantage is the wide selection of IDEs that can be used, including a rather robust option in Visual Studio.
3. **Storage Management**: The simple configuration for storage management allows for easy navigation on the back end. While owning a cloud server would be another expense to consider, it would be a great investment if the client ever has plans to expand in the future.
4. **Memory Management**: With Windows OS, you get 2 options, physical or virtual memory. For larger programs, virtual memory is a good option, and can handle the program effectively.
5. **Distributed Systems and Networks**: This is an area that would require attention with the Windows OS. Some issues that can arise are lagging, overloaded servers, and recurring queueing issues are just a few.
6. **Security**: With any OS, you would want to have security as something to always be thinking about. With Windows OS, there are features that can be utilized such as the Windows Defender antivirus program that runs on the client’s computer. It also allows for VPN services to help with protection of user information.